Framework for online simulation of soft robots with optimization-based inverse model

Christian Duriez, Eulalie Coevoet, Frédérick Largilliere, Thor Morales Bieze, Zhongkai Zhang, Mario Sanz Lopez, Bruno Carrez, Damien Marchal, Olivier Goury, Jérémie Dequidt

Framework for online simulation of soft robots with optimization-based inverse model

C. Duriez, E. Coevoet, F. Largilliere, T. Morales-Bieze, Z. Zhang,
M. Sanz-Lopez, B. Carrez, D. Marchal, O. Goury, J. Dequidt*.

***Abstract*— Soft robotics is an emerging field of robotics which requires computer-aided tools to simulate soft robots and provide models for their control. Until now, no unified software framework covering the different aspects exists. In this paper, we present such a framework from its theoretical foundations up to its implementation on top of SOFA, an open-source framework for deformable online simulation. The framework relies on continuum mechanics for modeling the robotic parts and boundary conditions like actuators or contacts using a unified representation based on Lagrange multipliers. It enables the digital robot to be simulated in its environment using a *direct* model. The model can also be inverted online using an optimization-based method which allows to control the physical robots in the task space. To demonstrate the effectiveness of the approach, we present various soft robots scenarios including ones where the robot is interacting with its environment. The software is freely available from https://project.inria.fr/softrobot/**

I. Introduction

Soft robotics raises interdisciplinary challenges involving material science, mechanical and electrical engineering, control theory, chemistry, physics, biology, computational mechanics and computer science. While the term *soft* is used, it actually means *non rigid* and is therefore employed for robots whose mechanical functioning relies on using deformable structures in a way similar to the biological world and organic materials. The use of deformable materials makes them very compliant, which provides natural key positive outcomes. Soft robots exhibit new types of functional capabilities that are complementary to traditional robotics. They can improve the safety of access to fragile parts of an environment by applying minimal pressure to its walls. Moreover, their large number of degrees of freedom combined with a redundant actuation can ease the manoeuvring through soft and confined spaces. This is particularly relevant for medical and surgical robotics [1], manipulation of fragile objects, domestic robotics with safer interactions with humans, arts and entertainment [2].

However, these outcomes often require a complex design. Building robots capable of complex tasks relies on having modeling and simulation tools [3], which is now a standard element in toolkits dedicated to rigid robotics. However, no such tool exists in soft robotics. The main reason is related to the motion of soft robots obtained through deformation of the structure rather than by articulations. Therefore, the behavior of soft robots should be modeled using deformable mechanics. Quoting [4]:

> "There exists well established theories as mechanics of continuous media. In robotics, we need to extract minimal models exploitable for analysis, for control, and to help direct goal-oriented design in particular toward control. In this respect, it will require a big effort to build generic modeling tools suited to soft robotics".

The presented work is our contribution to this *"big effort"*. The use of continuum mechanics raises several issues. No analytic solution exists in the general case and numerical methods, typically the finite element method (FEM), have to be used. This involves the discretization of the robot geometry which is not trivial (quality of the elements, trade-off between accuracy and computation time…). In addition, due to their natural compliance, soft robots are often used in contact with their environment, which increases the complexity of the modeling as well as the computational cost as identified in recent surveys about deformable robots [2], [5], [6], [7].

In this paper, we present a new software framework to model and simulate soft robots and their environment. The framework uses continuum mechanical modeling of soft materials combined with the Finite Element Method (FEM) for their numerical resolution. Boundary conditions are defined as constraints for both contacts and robot actuators. This framework unifies several of our previous works among which: the methodology of the inverse optimization to transfer the motion from task space to motion space presented in [8] with direct simulation [9], dynamic and quasi-static formulations [10]. It also provides contact management as in [11] and apparent stiffness control of the structure in case of redundant actuation [12].

The framework is implemented as a plugin for SOFA, an open-source toolkit geared towards interactive medical simulation. The motivations to use a medical simulation framework for robotic applications are numerous. Medical simulation and soft robotics make use of strongly deformable materials in complex arrangements. SOFA allows the simulation of such a complex arrangement and features many deformable models, several spatio-temporal integration schemes and accurate contacts management. It also interfaces many hardware sensors or haptic devices and finally can be run both offline and online. Using this plugin, we simulate soft robots from the state of the art [13], [14] as well as our own prototypes which include robots for grasping, navigating, handling objects or interacting with humans.

This paper is organized as follows: in Section II, we

*Authors are in *Defrost* team: INRIA, CNRS, University of Lille and Ecole Centrale de Lille, France. contact: christian.duriez@inria.fr

provide an overview of the modeling and simulation tools for soft robotics. Section III contains the theoretical foundation of our framework. Section IV explains how direct and inverse models are implemented. Section V contains implementation aspects related to the SOFA plugin and Section VI presents examples of soft robots modeled and simulated with our framework.

II. Related work

Soft robotics is a very recent and active field where researchers are actively exploring robots designs and their usages. One of the difficulty with soft robotics is that softness can be achieved with various approaches from soft materials like silicone [15], [16] , micro-structured materials [17] or specifically designed geometrical arrangement of rigid parts as with Tensegrity structures [18]. In addition to the material itself, actuation systems are very diverse with approaches including cables [13], pneumatics [19], [20], shape memory alloys [21] or chemical reaction [14].

In the field of rigid robotics, one can use either dedicated products like WorkspaceLt, RoboticSimulation, NI-Robotics RoboNaut or SimRobot or general purpose open-source software like Gazebo [3]. The cited tools rely on off-the-shelves simulation kernels such as Open Dynamics Engine, Bullets, NVidia PhysX or DART. These simulation kernels come from the video-game industry and are often focused on articulated-rigid bodies. They have been successfully used to model and simulate soft robots as in the NASA Tensegrity Robotics Toolkit [18] or in [22] with the use of PhysX to evaluate the candidate solutions of genetic algorithms. The video-game based simulation frameworks are fast and efficient to compute rigid-body simulations as well as some kind of soft bodies. They are also relatively easy to use as required background knowledge in physical modeling is reduced. The counterpart is that very few of them are capable of modeling physically realistic deformable materials.

When a realistic deformable material simulation is needed, tools from the structural and multi-physics analysis field, as Abaqus or ComSol, are an option. They rely on precise modeling formulations of continuous mechanics and some of them are capable to handle multi-physics. The cost for such capabilities is the slow computation speed and the fact that a good understanding of physical modeling is required. The consequence is that they are only usable for offline simulation of soft robots in combination with CAD software while designing the soft robotic parts [20]. Simpler alternatives exist such as Voxelyze. Presented in [23], it simulates soft materials undergoing large deformations and is associated with VoxCAD a GUI simplifying the editing of the robot. Voxelyze relies on voxels to represent the object. It is used in [24] to evaluate through simulation the walking capabilities of soft robots produced by genetic algorithms. In [25], the same authors added interaction between the robot and its environment. Nevertheless, with a voxel simulation, it is not possible to approximate some geometrical shapes without an exaggerated number of voxels which leads to an increased computation time. In addition, with Voxelyze, the mechanical model is using beam theory on the lattice supporting the voxels. Such an approach may not capture the continuous material deformation in a realistic manner.

Research in the field of surgical and biomedical simulation also developed simulation framework [26]. The interesting point of these frameworks is the focus on deformable objects and complex interactions. A tool like SOFA can simulate a large choice of mechanical models: from rigid-bodies or mass-spring to one implementing realistic hyperelastic material with FEM [27]. They can operate on a wide range of geometrical descriptions from 1D (curve) and 2D (surface mesh) to 3D (voxels, multi-resolution octrees [28] or hexahedral and tetrahedral mesh). They are capable of handling collisions and contacts precisely as well as to handle multi-physics behaviors [29], [30]. They are also capable of interacting with sensing hardware (Kinect, OptiTrack, LeapMotion) that are commonly used in robotics as well as with haptic devices [31]. A framework like SOFA can be considered as a *bridge* between video-game and structural analysis approaches and is chosen for this work.

In the following of this paper, we will present the plugin we realized for SOFA that is dedicated to soft robotics.

III. Modeling for real-time simulation

The theoretical foundations of our simulation framework for deformable objects are the ones of continuum mechanics for the material modeling, Lagrangian multipliers for constraints solving, and Signorini's law for contacts.

Let us start with the formulation given by the second law of Newton, that models the dynamic behavior of a body as:

$$\mathbb{M}(\mathbf{q})\dot{\mathbf{v}} = \mathbb{P}(t) - \mathbb{F}(\mathbf{q}, \mathbf{v}) + \mathbf{H}^T\lambda \tag{1}$$

where $\mathbf{q} \in \mathbb{R}^n$ is the vector of generalized degrees of freedom (for instance, displacement of the nodes of a mesh), $\mathbb{M}(\mathbf{q}) : \mathbb{R}^n \mapsto \mathcal{M}^{n\times n}$ is the inertia matrix, $\mathbf{v} = \dot{q} \in \mathbb{R}^n$ is the vector of velocity. $\mathbb{F}$ represents internal forces applied to the simulated object depending on the current state and $\mathbb{P}$ gathers known external forces. $\mathbf{H}^T$ is the matrix containing the constraint directions while $\lambda \in \mathbb{R}^n$ is the vector of Lagrange multipliers containing the constraint force intensities. In the following, we will present how these different terms can be computed.

A. Mechanical modeling

To compute $\mathbb{F}$, one needs to pick a deformation law. The underlying assumption is that all solids are deformable and the amount of deformation depends only on the external loads. This relationship between the loads and resulting deformations is the constitutive equation. A common equation, the Hooke's law, makes the assumption of linearity of material response to loads. Other laws exists to express nonlinear strain-stress relationship, plastic deformations, brittles or hysteresis behavior. Different laws have different computation costs and one has to carefully choose the law that fits best the needs and computation time constraints. Most of the time, we limit the deformation cases to purely elastic behavior: the robot goes back to its initial shape when

the actuation is released and the parameters of the materials are given by the Young modulus and the Poisson's ratio of the Hook's law. Different levels of complexity exist in the elastic deformation law which are: *small displacements, large displacements, large deformations* but for most of our robots, we rely on *large displacements* where a non-linear computation is performed to obtain the strain with a linear stress-strain relationship.

Depending on the constitutive equations and the geometrical representation, several possibilities exist in Sofa to model deformable materials. When dealing with 1D structures, one can use beam elements [32] or geometrical curves as in [33]. For 3D structures, there exist mass-spring models, co-rotational FEM [34], embedded deformable solids [28] as well as hyperelastic models to handle *large deformations* [27]. More concretely, each of these models can compute the $\mathbb{F}$ term in Equation 1.

B. Actuator constraint

In our framework, we handle the actuation by defining specific constraints with Lagrange multipliers on the boundary conditions of the deformable models.

Two types of actuators are considered in this work:

- Cable: when actuation is done by placing cables inside the structure of the robot to pull at certain points and create a deformation. The function $\boldsymbol{\delta}_a(\mathbf{x})$ measures the length of the cable which is modified by the actuation. $\boldsymbol{\lambda}_a$ is the force applied by the cable on the structure.
- Pneumatic: when actuation is done by exerting a variation of pressure on the surface of the deformable material. In such a case, $\boldsymbol{\delta}_a(\mathbf{x})$ is a measure of the volume of the cavity. $\boldsymbol{\lambda}_a$ is the uniform pressure inside the cavity.

It is possible to specify the behavior of the actuator either by assigning the value of $\boldsymbol{\lambda}_a$ or by setting the value of $\boldsymbol{\delta}_a(\mathbf{x})$ in the resolution process.

C. Contact constraint

When a potential contact on the robot has been detected, $\boldsymbol{\delta}_c(\mathbf{x})$ measures the shift between the robot and the obstacle at the contact point and $\boldsymbol{\lambda}_c$ is the contact force. In order to add the modeling of the environment, we need to deal with contact mechanics and find the value of $\boldsymbol{\lambda}_c$. For that, we will rely on a formulation of the complementarity problem using Signorini Conditions [11], [27]:

$$0 \leq \boldsymbol{\delta}_c \perp \boldsymbol{\lambda}_c \geq 0 \tag{2}$$

D. End effector and task space definition

It is possible to specify a constraint in the task space. It is particularly useful to obtain a direct or inverse model of the robot (see the following section). In this case, $\boldsymbol{\delta}_e(\mathbf{x})$ measures the shift along *x, y* and *z* between controlled point(s), which is (are) considered as effector and desired position(s) or trajectory.

For all constraint types, at each line i of the matrix $\mathbf{H}$, we have $\mathbf{H}_i = \frac{\partial \boldsymbol{\delta}_i(\mathbf{x})}{\partial \mathbf{x}}$.

IV. Numerical resolution

In this section, we describe how we integrate in time the equation of the dynamics (Eq. 1) and the numerical approaches used to solve the constraints.

A. Time integration or quasi-static formulation

We integrate equation 1 using a time-stepping implicit scheme (backward Euler) to have unconditional stability. Let us consider the time interval $[t_i, t_f]$ whose length is $h = t_f - t_i$:

$$\mathbb{M}(\mathbf{v}_f - \mathbf{v}_i) = h\left(\mathbb{P}(t_f) - \mathbb{F}(\mathbf{q}_f, \mathbf{v}_f)\right) + h\mathbf{H}^T\lambda \tag{3}$$

$$\mathbf{q}_f = \mathbf{q}_i + h\mathbf{v}_f \tag{4}$$

The internal forces $\mathbb{F}$ are a nonlinear function of the positions and the velocities. We then apply a Taylor series expansion to $\mathbb{F}$ and make the following first order approximation:

$$\mathbb{F}(\mathbf{q}_f, \mathbf{v}_f)) = \mathbb{F}\left(\mathbf{q}_i + d\mathbf{q}, \mathbf{v}_i + d\mathbf{v}\right) = \mathbf{f}_i + \frac{\delta \mathbb{F}}{\delta \mathbf{q}} d\mathbf{q} + \frac{\delta \mathbb{F}}{\delta \mathbf{v}} d\mathbf{v} \tag{5}$$

Using $d\mathbf{q} = \mathbf{q}_f - \mathbf{q}_i = h\mathbf{v}_f$ and $d\mathbf{v} = \mathbf{v}_f - \mathbf{v}_i$, we obtain:

$$\underbrace{\left(\mathbb{M} + h\frac{\delta \mathbb{F}}{\delta \mathbf{v}} + h^2\frac{\delta \mathbb{F}}{\delta \mathbf{q}}\right)}_{\mathbf{A}} \underbrace{d\mathbf{v}}_{\mathbf{dx}} = \underbrace{-h^2\frac{\delta \mathbb{F}}{\delta \mathbf{q}}\mathbf{v}_i - h\left(\mathbf{f}_i + \mathbf{p}_f\right)}_{\mathbf{b}} + h\mathbf{H}^T\lambda \tag{6}$$

where $\mathbf{p}_f$ is the value of the function $\mathbb{P}$ at time t_f. The only unknown values are the Lagrange multipliers λ ; their computation is detailed in Section IV-B. In the remainder of this section, we will refer to this system using the matrix $\mathbf{A}$ and the vector $\mathbf{b}$.

If the deformable robot is attached to the ground (like a manipulator) and its motion is performed at a low velocity, we can ignore the dynamic part (Eq. 1) and use a static formulation:

$$\mathbb{P} - \mathbb{F}\left(\mathbf{q}\right) + \mathbf{H}^T\lambda = 0 \tag{7}$$

Again, the Taylor series expansion (5) can be used to obtain a unique linearization per simulation step:

$$\underbrace{\frac{\delta \mathbb{F}}{\delta \mathbf{q}}}_{\mathbf{A}} \underbrace{d\mathbf{q}}_{\mathbf{dx}} = \underbrace{\mathbb{P} - \mathbf{f}_i}_{\mathbf{b}} + \mathbf{H}^T\lambda \tag{8}$$

We obtain a formulation similar to the dynamic case (Eq. 6) with $h = 1$.

B. Solving the constraints

From Equation 6 in dynamics or 8 in quasi-statics, the equation has two unknowns: $d\mathbf{x}$ which provides the motion of the degrees of freedom and $\lambda_\mathbf{i}$, which is the intensity of the actuators and contact loads. Consequently, the solving process will be executed in two steps.

The first step consists in obtaining a free configuration $\mathbf{q}_{\text{free}}$ of the robot that is found by solving Equation 8 while

considering that there is no actuation and no contact applied to the deformable structure.

$$\mathbf{A}d\mathbf{x}^{\text{free}} = \mathbf{b} \quad (9)$$

$$\mathbf{q}_{\text{free}} = \mathbf{q_i} + h(\mathbf{v_i} + d\mathbf{x}_{\text{free}}) \text{ (dynamic)} \quad (10)$$

$$\mathbf{q}_{\text{free}} = \mathbf{q_i} + d\mathbf{x}_{\text{free}} \text{ (quasi} - \text{static)} \quad (11)$$

To solve the linear equation (Eq. 9), we use a LDL^T factorization of the matrix $\mathbf{A}$. Given this new *free* position $\mathbf{q}_{\text{free}}$ for all the nodes of the mesh (i.e. position obtained without load on actuation or contact), we can evaluate the values of $\boldsymbol{\delta}_i^{\text{free}} = \boldsymbol{\delta}_i(\mathbf{q}_{\text{free}})$, defined in the previous section.

The second step is based on an optimization process that provides the value of $\boldsymbol{\lambda}$. In the following sections, we will define two cases of use: *direct and inverse modeling*. In both cases, the approach relies on an optimization process and its output is the value of the Lagrange multipliers. The size of matrix $\mathbf{A}$ is often very large so an optimization in the motion space would be computationally very expensive. To perform this optimization in real-time, we propose to project the problem in the constraint space using the Schur complement:

$$\boldsymbol{\delta}_i = \underbrace{h^2 \left[\mathbf{H}_i \mathbf{A}^{-1} \mathbf{H}_j^T\right]}_{\mathbf{W}_{ij}} \boldsymbol{\lambda_j} + \boldsymbol{\delta}_j^{\text{free}} \quad (12)$$

The physical meaning of this Schur complement is central in the method. $\mathbf{W}_{ij}$ provides a measure of the instantaneous mechanical coupling between the boundary conditions i and j, whether they correspond to an effector, an actuator or a contact. In practice, this projection allows to perform the optimization with the smallest possible number of equations.

It should be emphasized that one of the main difficulties is to compute $\mathbf{W}_{ij}$ in a fast manner. No precomputation is possible because the value changes at each iteration. But this type of projection problem is frequent when solving friction contact on deformable objects, thus several strategies are already implemented in SOFA [11], [27].

After solving the optimization process described in the two following subsections (Direct and Inverse modeling), we get the value of $\boldsymbol{\lambda}$, and we can compute the final configuration of the soft robot, at the end of each time step using:

$$d\mathbf{x} = d\mathbf{x}_{\text{free}} + h\mathbf{A}^{-1}\mathbf{H}^T\boldsymbol{\lambda} \quad (13)$$

Which provides the solution to equation 6 or 8.

a) Direct modeling of the robot in its environment: the inputs are the actuator values (either δ_a or λ_a) and the output is the displacement of the effector. When δ_a is the input, the optimization provides the values of λ_a as output. In case of contact, an additional output is the contact response λ_c (also found by optimization).

As explained above, using the operator $\mathbf{W}_{ea}$, we can get a measure of the mechanical coupling between effector(s) and actuator(s), and with $\mathbf{W}_{aa}$, the coupling between actuators. On a given configuration, $\mathbf{W}_{ea}$ provides a linearized relationship between the variation of displacement $\Delta\boldsymbol{\delta}_e$ created on the end-effector and the variation of the effort $\Delta\boldsymbol{\lambda}_a$ on the actuators. To get a direct kinematic link between actuators and effector point(s), we need to account for the mechanical coupling that can exist between actuators. This coupling is captured by $\mathbf{W}_{aa}$ that can be inverted if actuators are defined on independent degrees of freedoms. Consequently, we can get a kinematic link by rewriting Equation 12:

$$\Delta\boldsymbol{\delta}_e = \mathbf{W}_{ea}\mathbf{W}_{aa}^{-1}\Delta\boldsymbol{\delta}_a \quad (14)$$

This relationship provides (in the most condensed way) the displacement of the effector given the displacements of the actuators. Matrix $\mathbf{W}_{ea}\mathbf{W}_{aa}^{-1}$ is equivalent to a jacobian matrix for a standard, rigid robot. This corresponds to a local linearization provided by the FEM model on a given configuration and this relationship is only valid for *small variations* of $\Delta\boldsymbol{\delta}_a$, and in contactless cases.

b) Inverse modeling of the robot: the input is the desired position of the effector and the output is the force λ_a or the motion δ_a that needs to be applied on the actuators in order to minimize the distance with the effector position. λ_a is found by optimization and δ_a can be obtained using equations 12. We use this method in contactless cases.

The optimization consists in reducing the norm of $\boldsymbol{\delta}_e$ which actually measures the shift between the end-effector and its desired position. Thus, computing $min(\frac{1}{2}\boldsymbol{\delta}_e^T\boldsymbol{\delta}_e)$ can be done by setting a Quadratic Programming (QP) problem:

$$min\left(\frac{1}{2}\boldsymbol{\lambda_a}^T\mathbf{W}_{ea}^T\mathbf{W}_{ea}\boldsymbol{\lambda_a} + \boldsymbol{\lambda_a}^T\mathbf{W}_{ea}^T\boldsymbol{\delta}_e^{\text{free}}\right) \quad (15)$$

$$\begin{aligned} &subject\ to \text{ (course of actuators)}: \\ &\delta_{min} \leq \boldsymbol{\delta}_a = \mathbf{W}_{aa}\boldsymbol{\lambda_a} + \boldsymbol{\delta}_a^{\text{free}} \leq \delta_{max} \\ &and \text{ (case of unilateral effort actuation)}: \\ &\boldsymbol{\lambda_a} \geq 0 \end{aligned} \quad (16)$$

The use of a minimization allows to find a solution even when the desired position is out of the workspace of the robot. In such a case, the algorithm will find the point that minimizes the distance with the desired position while respecting the limits introduced for the stroke of the actuators.

The matrix of the QP, $\mathbf{W}_{ea}^T\mathbf{W}_{ea}$, is symmetric. If the number of actuators is equal or less than the size of the effector space, the matrix is also positive-definite. In such a case, the solution of the minimization is unique.

In the opposite case, i.e when the number of actuators is greater than the degrees of freedom of the effector points, the matrix of the QP is only semi-positive and the solution could be non-unique. In such a case, some QP algorithms are able to find one solution among all possible solutions [35]. In practice, we add to the cost function of the optimization a minimization of the deformation energy in the actuator space: The QP matrix is regularized by adding $\epsilon\mathbf{W}_{aa}$ (with ϵ chosen sufficiently small to keep a good accuracy on the effector motion).

V. Implementation

In the previous section, we have presented the theoretical foundation of our approach. We will now see more concretely how this translates in the SOFA plugin.

A. Concepts of the framework

In a way similar to Gazebo [3], SOFA has a scene-graph based simulation architecture. A scene contains the robot and its environment and is described in XML or with a Python script. SOFA is also a component based architecture. A robot is then an assembly of elementary components: some components are for rendering, others for contact or topology encoding, others for numerical integration or mechanical modeling. For simulation, the most important components are the *Mass* (that computes $\mathbb{M}$), the *Force Fields* ($\mathbb{P}(t)$, $\mathbb{F}$ and $\frac{\delta \mathbb{F}}{\delta \mathbf{q}}$)), the *MechanicalObject* (which stores the state vectors $\mathbf{q}$, $\mathbf{v}$, $d\mathbf{q}$).

B. Multi-model and mapping

SOFA also introduced multiple model representation. Multi-model means that in a simulated object a property can be represented in multiple ways. A good example is the shape of a robot. A low resolution tetrahedral mesh can be used for FEM while using a high resolution triangular mesh for the rendering and a low resolution triangle mesh for the contact management.

To connect the representations, SOFA introduces *Mappings*. Given the position of the degrees of freedom $\mathbf{q}$ of a representation, one can define a second representation with $\mathbf{p} = \mathbb{J}(\mathbf{q})$, where $\mathbb{J}$ is a possibly non-linear mapping function.

The strength of *Mappings* is that constraints on a representation can be transferred to a second representation. Using this tool, we can gather actuators, effectors and contacts from different representations to obtain a full system without having to change the implementation of the components. They are defined regardless of the geometrical dimension of the object (1D, 2D, 3D), geometrical representation (voxel, curve, octree, tetrahedral mesh) or mechanical model.

C. Soft robotic actuators

Using our framework, we have implemented simple actuator designs for cables and pressure.

For cable actuation, the function $\boldsymbol{\delta}_a(\mathbf{q})$ measures the length of the cable, depending on the position of the degrees of freedom $\mathbf{q}$ of the deformable model. To avoid re-coding the model of the cable for each type of degrees of freedom in $\mathbf{q}$, we define the path of the cable on a set of 3D points $(\mathbf{x}_1, ..., \mathbf{x}_n)$ that are mapped on the motion of $\mathbf{q}$ using a mapping $\mathbb{J}$. In practice, servomotors and pulleys are used to pull the cable. The rotation angle of the motor is either provided as a setpoint or computed using the inverse model. An Arduino® board pilots the motors online. The simulation sends the setpoints to the Arduino with a serial communication protocol.

In the case of pressure actuators, the function $\boldsymbol{\delta}_a(\mathbf{q})$ measures the volume of the cavity, depending also on the position of the degrees of freedom $\mathbf{q}$ of the deformable model. The surface pressure is applied via a *Mapping* to the underlying volumetric model. We then model and simulate the pneumatic actuator shown in Figure 1. In practice, electronically controlled pneumatic valves are used to apply the computed pressure into the cavities.

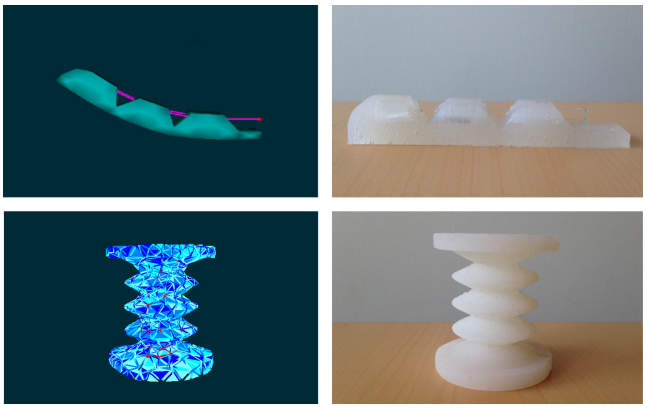

Fig. 1. Simulation (left) and real (right) cable actuators (top) and pressure actuators (bottom).

VI. Results & Validation

The methodology described in this paper allows for the simulation of soft robots in their environment in real-time. Moreover, the inverse model allows online control in the actuator space. The model and control of various robots with different geometric and mechanical characteristics, as well as different actuation schemes, are presented in this section. Table VI-A provides a quick overview of the results. The computation timing are based on a modern machine (Intel Core i5-4590 CPU). *This paper is accompanied by a video that allows to have a better understanding of the results.*

A. Graspers

In [36], the authors present an underactuated grasper design. This grasper is made of silicone and the actuation is done by three cables pulled by a single motor. We successfully modeled this robot and simulated it in real-time. The fingers are simulated using corotational FEM and the cube is a rigid body. Corotational FEM relies on a tetrahedral [1]mesh. Contact and friction modeling are added to the object to allow prehension. This design evolved into the grasper of Flexo (Subsect. VI-D) made of 3D printed plastic. A close approach was then used to model and simulate an octopus tentacle and the movement of the simulated tentacle was then compared to the physical one (Fig. 2).

B. Diamond: A platform made of silicone and actuated through cables.

The Diamond platform (Fig. 4) is made of a single piece of silicone. Four cables, pulling the structure, are connected to servomotors for actuation. For simulation, the silicone is modeled using FEM. The robot can then be controlled, either in the simulation or in the real world, from its endpoint thanks to our inverse simulation method. A stereo-vision system is used to track the position of points of interest in the real robots, which are then compared to those given by the simulation. The maximum positioning error obtained by

[1]For most of our robots, the volumetric meshes are generated directly in SOFA with the CGAL [37]

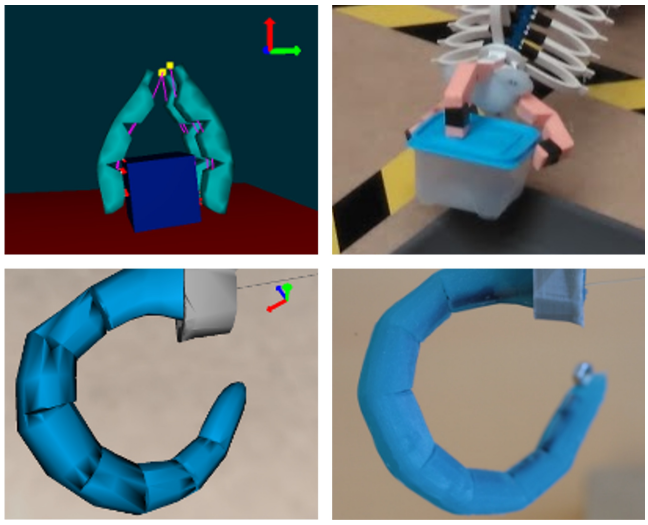

Fig. 2. Simulation of the soft grasper described in [36] (top left). Grasper of Flexo (top right). Comparison between simulated (bottom left) and real (bottom right) octopus tentacle based on a similar design.

Robot	Mat.	Act.	Cnt.	Ctrl.	Num.	Time(ms)
Grasper	S, P	C	Y	D	1422	5
Octoleg	S	C	Y	D	360	1.7
Diamond	S	C	N	I	4884	22
Fetch	S, P	C, P	N	I	12*	7.5
Flexo	P	C	N	I	912	4.5
Stifface	S	C	Y	I	2285	30
Sofia	S, P	C	Y	D	7818	100

Fig. 3. The different robots modeled and simulated with their Material (**S**ilicone, 3D Printed **P**lastic), Actuation (**C**able, **P**neumatic), Contacts (**Y**es, **N**o), Control (**D**irect or **I**nverse), Number of Degrees of Freedom and the computation Time (in ms) of one simulation step.
* *see paragraph* VI-C

the inverse model in open loop is 2,9mm, the mean error is 1,4mm. More details are given in [9] and [38].

C. Fetch: A pneumatic manipulator made of silicone.

The robotic part presented in Figure 5 is a differential pressure platform. Three cavities with a cylindrical accordion shape are inflated to provide an elongation that will extend or tilt the whole element. Several of these platforms can be stacked to increase the reachable space of the robot. The size of the FEM model of the robot (15456 degrees of freedom) would have prevented from real-time computation. Consequently, we have applied the model reduction method detailed in [10] to obtain a strong reduction of the number of degrees of freedom. The model reduction is based on the structure of the robot (continuum robot with rigid vertebrae). We end up with a model based on the degrees of freedom of the rigid vertebrae (here 12 degrees of freedom). The deformations and the pressure actuation model are *mapped* on these degrees of freedom to accelerate the computation.

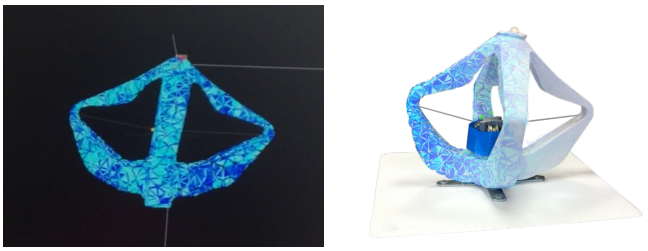

Fig. 4. Simulated version of the Diamond robot (left) and the physical one (right).

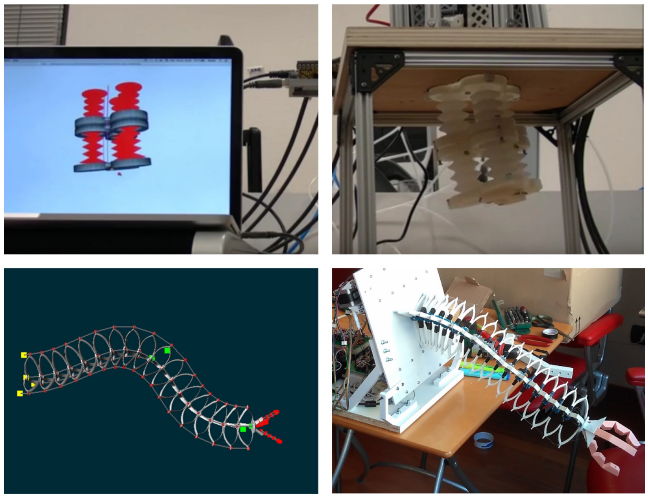

Fig. 5. Top: Control from inverse simulation of two Fetch platforms stacked for the Soft Robotic Toolkit 2015 competition. Bottom: Simulation of Flexo (left) and the real robot (right).

D. Flexo: A manipulator made of deformable material actuated with cables

The robot presented in Figure 5 is composed of several sections, each of which are made of 3 fork-rib shapes disposed each 120 degrees around the longitudinal direction. Branches have been modeled with beam elements.

E. Stifface: a soft robot acting as a human computer interface

Deformable robotics allows the creation of novel haptics interfaces with soft materials. We use our framework to simulate and control the Stifface interface. The device, detailed in [12], is made of silicone and aims to render different apparent stiffness to a user exploring a virtual surface. The silicone is modeled with hyperelastic Neohookean FEM and the cable actuation system is reproduced in the same configuration as in the real device.

F. Sofia: a walking robot

The figure (Fig. 7) presents Sofia, a walking robot. It is made of 6 silicone legs. Its body is an assembly of 3D printed ABS parts with a structural pattern that increases its flexibility. There are 6 servomotors coupled through a crankshaft to the legs, as well as 6 others actuating the structure through cables. The robot was modeled and its

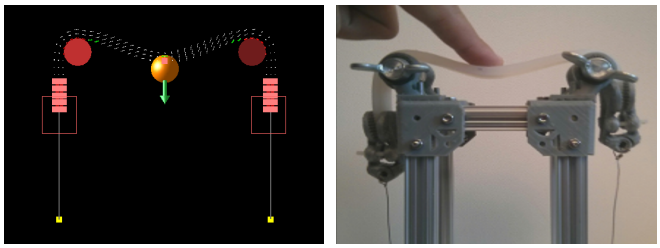

Fig. 6. Simulation of the Stifface interface (left) and real prototype (right).

walking motion was simulated in a virtual environment before going to the real field.

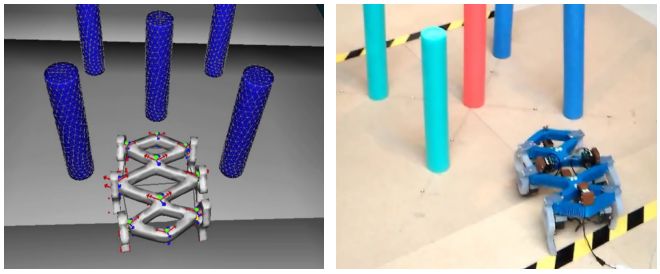

Fig. 7. The Sofia walking robot simulated in a virtual world (left) and in real during the Robosoft Grand Challenge 2016 (right).

VII. CONCLUSIONS

This paper presents the mathematical basis as well as a software framework that targets the design, simulation and control of soft robots. This framework relies on a mechanical representation of the robot, its actuators, effector and possible contacts with the environment. Using equations from continuum mechanics, the motion of the robot can be simulated. Moreover, the environment can be taken into account through its mechanical representation in the simulation. An inverse problem optimization automatically computes the actuation to obtain control in the task space. Real-time performance is obtained in both cases (direct and inverse) using relatively coarse meshes. The capabilities of this framework are illustrated with several examples and we show that a reasonable accuracy between simulated and real soft robots can be obtained.

The modularity of the framework encourages many extensions. For instance, future works may include adding more complex mechanical laws, adding robust control laws or designing complex and dynamic environments. This will increase the computational footprint of the simulation whereas the short computation time needs to be maintained in order to do online control of the robot. Therefore, advanced numerical methods such as reduced-order modeling or dedicated solvers should be considered to achieve sufficient accuracy without increasing the computation cost of the simulation.

REFERENCES

[1] M. Cianchetti, T. Ranzani, G. Gerboni, T. Nanayakkara, K. Althoefer, P. Dasgupta, and A. Menciassi, "Soft robotics technologies to address shortcomings in today's minimally invasive surgery: The stiff-flop approach," *Soft robotics*, vol. 1, no. 2, 2014.

[2] H. Lipson, "Challenges and opportunities for design, simulation, and fabrication of soft robots," *Journal of soft robotics*, vol. 1, no. 1, 2014.

[3] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *Proceeding of the conference on Intelligent Robots and Systems. (IROS).*, vol. 3, 2004.

[4] "First robosoft working paper," Future Emerging Technology (FET) on Soft-Robot, Tech. Rep., 2014.

[5] D. Trivedi, C. D. Rahn, W. M. Kier, and I. D. Walker, "Soft robotics: Biological inspiration, state of the art, and future research," *Applied bionics and biomechanics*, vol. 5, no. 3, 2008.

[6] S. Kim, C. Laschi, and B. Trimmer, "Soft robotics: A bioinspired evolution in robotics," *Trends in biotechnology*, vol. 31, no. 5, 2013.

[7] D. Rus and M. T. Tolley, "Design, fabrication and control of soft robots," *Nature*, vol. 521, no. 7553, 2015.

[8] E. Coevoet, N. Reynaert, E. Lartigau, L. Schiappacasse, J. Dequidt, and C. Duriez, "Introducing interactive inverse FEM simulation and its application for adaptive radiotherapy," in *Proceeding of the Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2014.

[9] C. Duriez, "Control of Elastic Soft Robots based on Real-Time Finite Element Method," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

[10] J. Bosman, T. M. Bieze, O. Lakhal, M. Sanz, R. Merzouki, and C. Duriez, "Domain decomposition approach for fem quasistatic modeling and control of continuum robots with rigid vertebras," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

[11] H. Courtecuisse, J. Allard, P. Kerfriden, S. P. Bordas, S. Cotin, and C. Duriez, "Real-time simulation of contact and cutting of heterogeneous soft-tissues," *Medical image analysis*, vol. 18, no. 2, 2014.

[12] F. Largilliere, E. Coevoet, M. Sanz-Lopez, L. Grisoni, and C. Duriez, "Stiffness rendering on soft tangible devices controlled through inverse fem simulation," in *Proceedings of International Conference on Intelligent Robots and Systems (IROS)*, 2016.

[13] T. Hassan, M. Manti, G. Passetti, N. d'Elia, M. Cianchetti, and C. Laschi, "Design and development of a bio-inspired, under-actuated soft gripper," in *Proceedings of the Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 2015.

[14] B. Mosadegh, P. Polygerinos, C. Keplinger, S. Wennstedt, R. F. Shepherd, U. Gupta, J. Shim, K. Bertoldi, C. J. Walsh, and G. M. Whitesides, "Pneumatic networks for soft robotics that actuate rapidly," *Advanced functional materials*, vol. 24, 2014.

[15] M. T. Tolley, R. F. Shepherd, B. Mosadegh, K. C. Galloway, M. Wehner, M. Karpelson, R. J. Wood, and G. M. Whitesides, "A resilient, untethered soft robot," *Soft robotics*, vol. 1, no. 1, 2014.

[16] R. V. Martinez, J. L. Branch, C. R. Fish, L. Jin, R. F. Shepherd, R. M. D. Nunes, Z. Suo, and G. M. Whitesides, "Robotic tentacles with three-dimensional mobility based on flexible elastomers," *Advanced materials*, vol. 25, no. 2, 2013.

[17] C. Schumacher, B. Bickel, J. Rys, S. Marschner, C. Daraio, and M. Gross, "Microstructures to control elasticity in 3d printing," *ACM Trans. Graph.*, vol. 34, no. 4, 2015.

[18] K. Caluwaerts, J. Despraz, A. Işçen, A. P. Sabelhaus, J. Bruce, B. Schrauwen, and V. SunSpiral, "Design and control of compliant tensegrity robots through simulation and hardware validation," *Journal of the royal society interface*, vol. 11, no. 98, 2014.

[19] R. F. Shepherd, F. Ilievski, W. Choi, S. A. Morin, A. A. Stokes, A. D. Mazzeo, X. Chen, M. Wang, and G. M. Whitesides, "Multigait soft robot," *Proceedings of the national academy of sciences*, vol. 108, no. 51, 2011.

[20] R. V. Martinez, C. R. Fish, X. Chen, and G. M. Whitesides, "Elastomeric origami: Programmable paper-elastomer composites as pneumatic actuators," *Advanced functional materials*, vol. 22, no. 7, 2012.

[21] A. Villanueva, C. Smith, and S. Priya, "A biomimetic robotic jellyfish (robojelly) actuated by shape memory alloy composite actuators," *Bioinspiration & biomimetics*, vol. 6, no. 3, 2011.

[22] J. Rieffel, F. Saunders, S. Nadimpalli, H. Zhou, S. Hassoun, J. Rife, and B. Trimmer, "Evolving soft robotic locomotion in physx," in *Proceedings of the 11th annual conference companion on genetic and evolutionary computation conference: Late breaking papers*, 2009.

[23] J. Hiller and H. Lipson, "Dynamic Simulation of Soft Multimaterial 3D-Printed Objects," *Soft robotics*, vol. 1, no. 1, 2014.

[24] N. Cheney, R. MacCurdy, J. Clune, and H. Lipson, "Unshackling evolution: Evolving soft robots with multiple materials and a powerful generative encoding," *SIGEVOlution*, vol. 7, no. 1, 2014.

[25] N. Cheney, J. Bongard, and H. Lipson, "Evolving soft robots in tight spaces," in *Proceedings of the conference on genetic and evolutionary computation*, 2015.

[26] Y. Payan, *Soft tissue biomechanical modeling for computer assisted surgery*. Springer, 2012, vol. 11.

[27] F. Faure, C. Duriez, H. Delingette, J. Allard, B. Gilles, S. Marchesseau, H. Talbot, H. Courtecuisse, G. Bousquet, I. Peterlik, and S. Cotin, "Sofa: A multi-model framework for interactive physical simulation," in *Soft tissue biomechanical modeling for computer assisted surgery*. 2012.

[28] M. Nesme, P. G. Kry, L. Jeřábková, and F. Faure, "Preserving topology and elasticity for embedded deformable models," *ACM Trans. Graph.*, vol. 28, no. 3, 2009.

[29] H. Talbot, S. Marchesseau, C. Duriez, M. Sermesant, S. Cotin, and H. Delingette, "Towards an interactive electromechanical model of the heart," *Interface focus royal society*, 2012.

[30] H. Talbot, F. Roy, and S. Cotin, "Augmented Reality for Cryoablation Procedures," in *SIGGRAPH*, 2015.

[31] I. Peterlik, M. Nouicer, C. Duriez, S. Cotin, and A. Kheddar, "Constraint-based haptic rendering of multirate compliant mechanisms," *IEEE Transactions on Haptics*, vol. 4, no. 3, 2011.

[32] C. Duriez, S. Cotin, J. Lenoir, and P. Neumann, "New approaches to catheter navigation for interventional radiology simulation," *Computer aided surgery*, vol. 11, no. 6, 2006.

[33] A. Theetten, L. Grisoni, C. Duriez, and X. Merlhiot, "Quasi-dynamic splines," in *Proceedings of the ACM Symposium on Solid and Physical Modeling*, 2007.

[34] C. A. Felippa, "A systematic approach to the element-independent corotational dynamics of finite elements," *Center for Aerospace Structures Document Number CU-CAS-00-03, College of Engineering, University of Colorado*, 2000.

[35] F. Sha, L. K. Saul, and D. D. Lee, "Multiplicative updates for nonnegative quadratic programming in support vector machines," in *Advances in neural information processing systems*, 2002.

[36] T. Hassan, M. Manti, G. Passetti, N. d'Elia, M. Cianchetti, and C. Laschi, "Design and development of a bio-inspired, under-actuated soft gripper," in *Proceeding of the Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, IEEE, 2015.

[37] A. Fabri and S. Pion, "Cgal: The computational geometry algorithms library," in *Proceedings of the international conference on advances in geographic information systems (icagis)*, 2009.

[38] F. Largilliere, V. Verona, E. Coevoet, M. Sanz-Lopez, J. Dequidt, and C. Duriez, "Real-time control of soft-robots using asynchronous finite element modeling," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015.